CHƯƠNG IV: CÁC HỆ MÃ MẬT KHÓA CÔNG KHAI

Trong các hệ mã mật khóa bí mật nếu chúng ta biết khóa và hàm mã hóa chúng ta có thể tìm được khóa và hàm giải mã một cách nhanh chóng (thời gian đa thức).

Một hệ mã mật khóa bí mật là một hệ mã mật mà tất cả mọi người đều biết hàm mã hóa và khóa mã hóa nhưng không tồn tại một thuật toán thời gian đa thức để có thể tính được khóa giải mã từ các thông tin đó.

1. Khái niệm hệ mã mật khóa công khai

Các hệ mã được trình bày trong các chương trước được gọi là các hệ mã khóa bí mật, khóa đối xứng, hay các hệ mã truyền thống (conventional).

Các hệ mã này có các điểm yếu sau đây:

- Nếu số lượng người sử dụng lớn thì số khóa sẽ tăng rất nhanh, chẳng hạn với n người sử dụng thì số khóa sẽ là n *(n-1)/2 do đó rất khó quản lý, phức tạp và không an toàn.
- Dựa trên các hệ mã này không thể xây dựng các khái niệm và dịch vụ như chữ ký điện tử, dịch vụ xác thực hóa người dùng cho các ứng dụng thương mại điện tử.

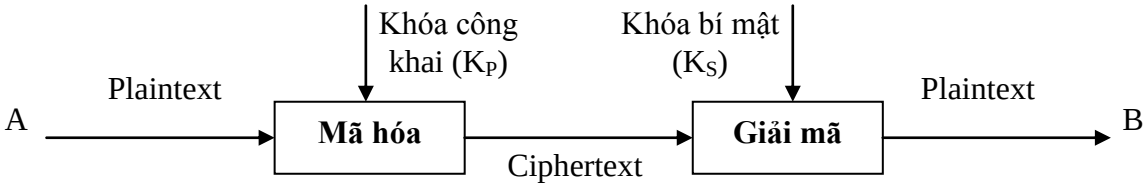
Vào năm 1975 Diffie và Hellman trong một công trình của mình (một bài báo) đã đề xuất ra các ý tưởng cho phép xây dựng lên các hệ mã hoạt động theo các nguyên tắc mới gắn liền với các bên truyền tin chứ không gắn với các cặp truyền tin.

Nguyên tắc hoạt động của các hệ mã là mỗi bên tham gia truyền tin sẽ có 2 khóa, một khóa gọi là khóa bí mật và một khóa được gọi là khóa công khai. Khóa bí mật là khóa dùng để giải mã và được giữ bí mật (K_S), khóa công khai là khóa dùng để sinh mã được công khai hóa để bất cứ ai cũng có thể sử dụng khóa này gửi tin cho người chủ của hệ mã (K_P). Ngày nay chúng ta có thể thấy rất rõ nguyên tắc này trong việc gửi email, mọi người đều có thể gửi email tới một địa chỉ email nào đó, nhưng chỉ có người chủ sở hữu của địa chỉ email đó mới có thể đọc được nội dung của bức thư, còn những người khác thì không. Với các hệ mã khóa công khai việc phân phối khóa sẽ trở nên dễ dàng hơn qua các kênh cung cấp khóa công cộng, số lượng khóa hệ thống quản lý cũng sẽ ít hơn (là n khóa cho n người dùng). Các dịch vụ mới như chữ ký điện tử, thỏa thuận khóa cũng được xây dựng dựa trên các hệ mã này.

Các yêu cầu của loại hệ mã này:

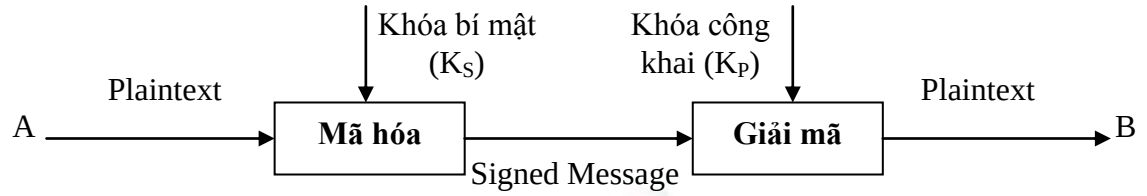
- Việc sinh K_P, K_S phải dễ dàng
- Việc tính $E(K_P, M)$ là dễ dàng
- Nếu có $C = E(K_P, M)$ và K_S thì việc tìm bản rõ cũng là dễ
- Nếu biết K_P thì việc dò tìm K_S là khó
- Việc khôi phục bản rõ từ bản mã là rất khó

Khi A muốn truyền tin cho B, A sẽ sử dụng khóa K_P của B để mã hóa tin tức và truyền bản mã tới cho B, B sẽ sử dụng khóa bí mật của mình để giải mã và đọc tin:



Hình 4.1: Mô hình sử dụng 1 của các hệ mã khóa công khai PKC

Ciphertext = $E(K_P,Plaintext)$,Plantext = $D(K_S, E(K_P,Plaintext))$ (1)



Hình 4.2: Mô hình sử dụng 2 của các hệ mã khóa công khai PKC

Ciphertext = $D(K_S, Plaintext)$, Plaintext = $E(K_P, D(K_S, Plaintext))$ (2)

Mô hình (2) được sử dụng c ho các hệ chữ ký điện tử còn mô hình (1) được sử dụng cho các hệ mã mật . Các hệ mã này được gọi là các hệ mã khóa công khai PKC (Public Key Cryptosystems) hay các hệ mã bất đối xứng (Asymmetric Encryption Scheme).

2. Nguyên tắc cấu tạo của các hệ mã mật khóa công khai

Các hệ mã khóa công khai được xây dựng dựa trên các hàm được gọi là các hàm 1 phía hay hàm 1 chiều (one–way functions).

Hàm một chiều $f : X \rightarrow Y$ làm một hàm mà nếu biết $x \in X$ ta có thể dễ dàng tính được $y = f(x)$. Nhưng với y bất kỳ $\in Y$ việc tìm $x \in X$ sao cho $y = f(x)$ là khó. Có nghĩa là việc tìm hàm ngược f^{-1} là rất khó.

Ví dụ nếu chúng ta có các số nguyên tố $P_1, P_2, ..., P_n$ thì việc tính $N = P_1 * P_2 * ... * P_n$ là dễ nhưng nếu có N thì việc phân tích ngược lại là một bài toán khó với N lớn.

Để thuận tiện các hàm một phía được sử dụng trong các hệ mã PKC thường được trang bị các cửa bẫy (trapdoor) giúp cho việc tìm x thỏa mã $y = f(x)$ là dễ dàng nếu chúng ta biết được cửa bẫy này.

Hàm của bẫy (trapdoor function): là một hàm một chiều trong đó việc tính f^{-1} là rất nhanh khi chúng ta biết được cửa bẫy của hàm . Ví dụ việc tìm nghiệm của bài toán xếp balô 0/1 trong hệ mã xếp balô Knapsack mà chúng ta sẽ học trong phần tiếp theo là một hàm một phía (việc mã hóa rất nhanh và dễ dàng nhưng tìm vectơ nghiệm tương ứng là khó) nhưng nếu ta biết cửa b ẫy (Vectơ xếp balô siêu tăng A') thì việc giải bài toán lại rất dễ dàng.

3. Một số hệ mã khóa công khai

3.1. Hệ mã knapsack

Bài toán xếp ba lô tổng quát:

Cho M, N và A_1, A_2, ..., A_N là các số nguyên dương tìm các số x_i không âm sao cho:

$$\mathbf{M} = \sum_{i=1}^{N} x_i * A_i$$

Vecto A = (A_1, A_2, ..., A_N) được gọi là vecto xếp balô còn vectơ X = (x_1, x_2, …, x_N) là vectơ nghiệm.

Một trường hợp riêng đáng quan tâm của bài toán xếp ba lô tổng quát là trường hợp mà $x_i \in \{0, 1\}$. Khi đó ta có bài toán xếp ba lô 0, 1.

Vecto xếp ba lô siêu tăng : Trong trường hợp vecto (A_1, A_2, ..., A_N) được sắp lại thành (A'_1, A'_2, ..., A'_N) sao cho:

$\forall$ i ta có: $\sum_{j<i} A'_j < A'_i$ thì vecto (A_1, A_2, ..., A_N) được gọi là vecto xếp balo siêu tăng.

Khi (A_1, A_2, ..., A_N) là một vecto xếp balo siêu tăng ta có ngay tính chất: M >= A'_i $\forall$ i. Do đó việc giải bài toán xếp ba lô 0/1 trở nên dễ dàng hơn rất nhiều.

Hệ mã knapsack do Merkle và Hellman đưa ra vào năm 1978.

Cách xây dựng:

1. Chọn 1 vecto siêu tăng A' = (a'_1, a'_2, ..., a'_N), chọn 1 số M > 2 * a'_N, chọn ngẫu nhiên 1 số u < M và (u, M) = 1
2. Xây dựng Vecto A = (a_1, a_2, ..., a_N) trong đó a_i = (a'_i * u) mod M
3. Khóa: K_P = (A, M), K_S = (u, u^{-1})
4. Không gian các bản rõ là không gian mọi dãy N bit

   P = (x_1, x_2, ..., x_n).

Mã hóa: C = ($\sum_{i=1}^{N} a_i * x_i$)mod M

Giải mã: tính C' = C * u^{-1} mod M sau đó giải bài toán xếp ba lô 0/1 với A', C' từ đó tìm được P = (x_1, x_2, ..., x_n).

Ví dụ 1: Cho hệ mã Knapsack có A' = (2, 3, 6, 12, 25), N = 5, M = 53, u = 46, u^{-1} = 15.

a) Hãy tìm các khóa của hệ mã trên

b) Mã hóa và giải mã bản mã tương ứng của bản rõ M = 01001.

3.2. Hệ mã RSA

Hệ mã RSA được đặt tên dựa theo các chữ cái đầu của 3 tác giả của hệ mã là Rivest, Shamir và Adleman. Đây là thuật toán mã hóa nổi tiếng nhất và cũng là thuật toán được ứng dụng thực tế nhất.

Để cài đặt RSA ban đầu mỗi người dùng sinh khóa công khai và khóa bí mật của mình bằng cách:

- chọn hai số nguyên tố lớn ngẫu nhiên (cỡ gần 100 chữ số) khác nhau p và q
- tính N = p*q
- chọn một số e nhỏ hơn N và (e, ∅(N)) = 1, e được gọi là số mũ lập mã
- tìm phần tử ngược của e trên vành module ∅(N), d là số mũ giải mã
- khóa công khai là K_P = (e, N)
- khóa bí mật là $K_S = K^{-1}_P$ = (d, p, q)

Việc thiết lập khóa này được thực hiện 1 lần khi một người dùng thiết lập (thay thế) khóa công khai của họ. Mũ e thường là khá nhỏ (để mã hóa nhanh), và phải là nguyên tố cùng nhau với ∅(N). Các giá trị thường được chọn cho e là 3 hoặc $2^{16} – 1 = 65535$. Tuy nhiên khi e nhỏ thì d sẽ tương đối lớn . Khoá bí mật là (d, p, q). Các số p và q thường có giá trị xấp xỉ nhau nhưng không được bằng nhau . Chú ý là việc để lộ một trong các thành phần trên sẽ làm cho hệ mã hóa trở thành không an toàn.

Sử dụng RSA

- để mã hóa một thông điệp M: $C = M^e \pmod N$ (0<= M < N)
- giải mã: $M = C^d \pmod N$

Thuật toán mã hóa RSA làm việc được bởi vì nó dựa trên cơ sở toán học là sự tổng quát định lý Ferma nhỏ của Ơclit: $X^{\varnothing(N)} = 1 \pmod N$. Trong thuật toán RSA chúng ta chọn e và d là nghịch đảo của nhau trên vành $Z_{\varnothing(N)}$ với e được chọn trước.

Do đó chúng ta sẽ có $e.d \equiv 1 \bmod \varnothing(N)$, suy ra:

$M = C^d = M^{e.d} = M^{1+q.\varnothing(N)} = M . (M^{\varnothing(N)})^q = M \bmod N$

Công thức này đảm bảo việc giải mã sẽ cho kết quả đúng là bản rõ ban đầu (chú ý là điều này chỉ đúng khi p khác q).

Ví dụ 1: Cho hệ mã RSA có N = p*q = 11 * 47 = 517, e = 3.

- Hãy tìm các khóa công khai và bí mật của hệ mã trên
- Mã hóa bản rõ M = 26.

Đầu tiên ta tính được ∅(N) = 460 = 10 * 46, do (3,460) = 1 nên áp dụng thuật toán Ơclit mở rộng ta tìm được d = 307.

Vậy khóa công khai của hệ mã K_P = (e, N) = (3, 517), khóa bí mật là K_S = (d, p, q) = (307, 11, 47).

Mã hóa M = 26 ta có $C = M^e \bmod N = 26^3 \bmod 517 = 515$.

Độ an toàn của RSA

Độ an toàn của RSA phụ thuộc vào độ khó của việc tính ∅(N) và điều này đòi hỏi chúng ta cần phân tích N ra thừa số nguyên tố. Thuật toán phân tích số nguyên tố hiệu quả nhất hiện nay là Brent-Pollard, chúng ta hãy xem xét bảng thống kê sau để thấy được tốc độ hoạt động của nó:

Số chữ số trong hệ thập phân của N	Số các thao tác Bit để phân tích N

20	7.20e+03
40	3.11e+06
60	4.63e+08
80	3.72e+10
100	1.97e+12
120	7.69e+13
140	2.35e+15
160	5.92e+16
180	1.26e+18
200	2.36e+19

Bảng 4.1: Tốc độ của thuật toán Brent-Pollard

Các nghiên cứu về vấn đề phân tích các số nguyên lớn hiện nay tiến triển rất chậm, các tiến bộ lớn nhất cũng chỉ là các cải tiến về thuật toán và có thể nói rằng trừ khi có các đột phá trong việc phân tích các số 1024 bit, RSA là an toàn trong thời điểm hiện nay.

Các nhà mật mã học phát minh ra hệ mã RSA đã đưa ra một giải thưởng trị giá 100 $ vào năm 1977. Đó là một hệ mã với số N có 129 chữ số, thách thức này đã được phá.

Trên thực tế để cài đặt RSA cần phải thực hiện các thao tác modulo với các số 300 chữ số (hay 1024 bit) mà hiện nay các máy tính mới chỉ thao tác với các số nguyên 64 bit, điều này dẫn đến nhu cầu cần các thư viện số học nhân chính xác để làm việc với các số nguyên lớn này. Ngoài ra việc sử dụng RSA cần tới các số nguyên tố lớn nên chúng ta cũng phải có một cơ sở dữ liệu các số nguyên tố.

Để tăng tốc cho RSA chúng ta có thể sử dụng một số phương pháp khác chẳng hạn như cải tiến các phép tính toán nhân hai số lớn hoặc tăng tốc việc tìm bản mã, bản rõ.

Đối với phép nhân 2 số n bit thông thường chúng ta cần thực hiện $O(n^2)$ phép tính bit. Thuật toán nhân các số nguyên Schonhage – Strassen cho phép chúng ta thực hiện phép nhân 2 số với độ phức tạp là O(n log n) với các bước như sau:

- Chia mỗi số nguyên thành các khối, sử dụng các khối này như các hệ số của một đa thức.
- Tính các đa thức này tại một số các điểm thích hợp, và nhân các kết quả thu được.
- Nội suy các kết quả này hình thành các hệ số của đa thức tích
- Kết hợp các hệ số để hình thành nên tích của hai số ban đầu
- Biến đổi Fourier rời rạc, và lý thuyết chặp có thể được sử dụng để tăng tốc độ của quá trình nội suy.

Một cách khác nữa để tăng tốc việc nhân các số lớn trong hệ mã RSA là sử dụng các phần cứng chuyên dụng với các thuật toán song song.

Như đã trình bày ở phần trước khi mã hóa chúng ta thường chọn e nhỏ để đẩy nhanh quá trình mã hóa nhưng điều này cũng đồng nghĩa là việc giải mã sẽ chậm do số mũ lớn. Một cải tiến đáng kể trong tốc độ giải mã RSA có thể nhận được bằng cách sử dụng định lý phần dư Trung Hoa làm việc với modulo p và q tương ứng thay vì N. Vì p và q chỉ bằng một nửa của N nên tính toán sẽ nhanh hơn nhiều.

Định lý phần dư Trung Hoa được sử dụng trong RSA bằng cách tạo ra hai phương trình từ việc giải mã $M = C^d \pmod N$ như sau:

$M_1 = M \bmod p = (C \bmod p)^{d \bmod (p-1)}$

$M_2 = M \bmod q = (C \bmod q)^{d \bmod (q-1)}$

Sau đó ta giải hệ:

$M = M_1 \bmod p$

$M = M_2 \bmod q$

Hệ này có nghiệm duy nhất theo định lý phần dư Trung Hoa

$M = [(M_2 + q - M_1)u \bmod q]\, p + M_1$

Trong đó $p.u \bmod q = 1$

Việc sử dụng định lý phần dư Trung Hoa là một phương pháp được sử dụng rộng rãi và phổ biến để tăng tốc độ giải mã của RSA.

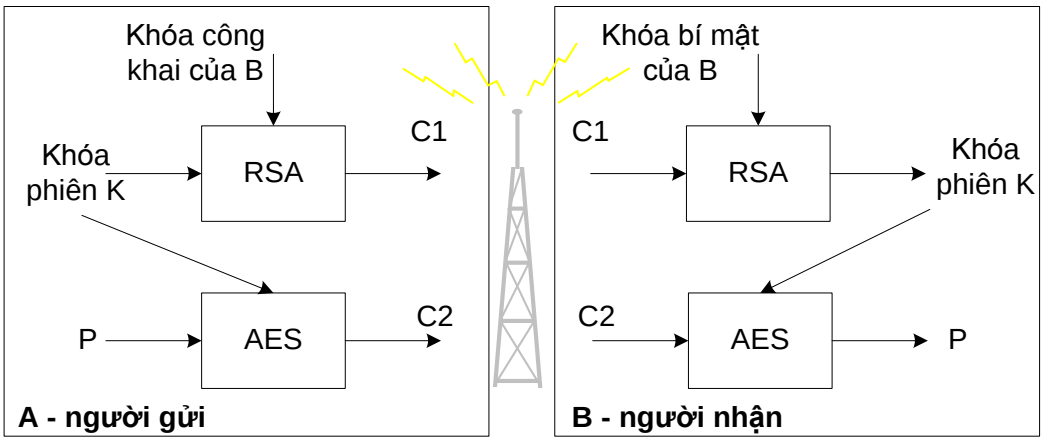
Hiện tượng lộ bản rõ

Một hiện tượng cần lưu ý khi sử dụng các hệ mã RSA là hiện tượng lộ bản rõ . Ta hãy xét hệ mã RSA có N = p*q = 5*7, e = 17, khi đó với M = 6 ta có $C = 6^{17} \bmod N = 6$.

Tương tự với hệ mã RSA có N = p*q = 109*97, e = 865, với mọi M ta đều có $M^e \bmod N = M$.

Theo tính toán thì với một hệ mã RSA có N = p*q và e bất kỳ, số lượng bản rõ sẽ bị lộ khi mã hóa sẽ là (1 + (e-1, p-1))*(1 + (e-1, q-1)).

Trong số các hệ mã khóa công khai thì có lẽ hệ mã RSA (cho tới thời điểm hiện tại) là hệ mã được sử dụng rộng rãi nhất.Tuy nhiên do khi làm việc với dữ liệu đầu vào (thông điệp mã hóa , bản rõ) lớn thì khối lượng tính toán rất lớn nên trên thực tế người ta hay dùng hệ mã này để mã hóa các dữ liệu có kích thước nhỏ , hoặc có yêu cầu bảo mật cao , chẳng hạn như các khóa phiên (session key) trong các phiên truyền tin. Khi đó hệ mã RSA sẽ được sử dụng kết hợp với một hệ mã khối khác , chẳng hạn như AES , theo mô hình lai ghép như sau:



Hình 4.3: Mô hình ứng dụng lai ghép RSA với các hệ mã khối

3.3. Hệ mã El Gamal

Hệ mã El Gamal là một biến thể của sơ đồ phân phối khoá Diffie – Hellman. Hệ mã này được El Gamal đưa ra vào năm 1985. Giống như sơ đồ phân phối khóa Diffie – Hellman tính an toàn của nó dựa trên tính khó giải của bài toán logarit rời rạc. Nhược điểm chính của nó là kích thước thông tin sau khi mã hóa gửi đi sẽ tăng gấp đôi so với thông tin gốc.

Tuy nhiên so với RSA, El Gamal không có nhiều rắc rối về vấn đề bản quyền sử dụng.

Ban đầu người ta sẽ chọn một số nguyên tố lớn p và hai số nguyên tuỳ ý nhỏ hơn p là a (a là một phần tử nguyên thủy của Z^*_P) và x (x là của người nhận, bí mật) sau đó tính:

$y = a^x \bmod p$

Để mã hóa một thông điệp M (là một số nguyên trên Z_P) thành bản mã C người gửi chọn một số ngẫu nhiên k nhỏ hơn p và tính khóa mã hóa K:

$K = y^k \bmod p$

Sau đó tính cặp bản mã:

- $C_1 = a^k \bmod p$
- $C_2 = K.M \bmod p$

Và gửi bản mã $C = (C_1, C_2)$ đi (chú ý là sau đó k sẽ bị huỷ).

Để giải mã thông điệp đầu tiên ta cần tính lại khóa mã hóa thông điệp K:

$K = C_1^x \bmod p = a^{k.x} \bmod p$

Sau đó tính M bằng cách giải phương trình sau đây:

$M = C_2 . K^{-1} \bmod p$

Việc giải mã bao gồm việc tính lại khóa tạm thời K (rất giống với mô hình của Diffie – Hellman đưa ra). Khóa công khai của hệ mã là (p, a, y), khóa bí mật là x.

Ví dụ: Cho hệ mã El Gamal có P = 97, a = 5, x = 58.

- Tìm khóa của hệ mã trên.
- Mã hóa bản rõ M = 3 với k được chọn bằng 36.

Trước hết ta tính $y = 5^{58} \bmod 97 = 44$, từ đó suy ra $K_P = (P, a, y) = (97, 5, 44)$ và $K_S = (58)$.

Để mã hóa thông điệp M = 3 ta tính khóa $K = 44^{36} \bmod 97 = 75$ sau đó tính:

- $C_1 = 5^{36} = 50 \bmod 97$
- $C_2 = 75.3 \bmod 97 = 31 \bmod 97$

Vậy bản mã thu được là C = (50, 31).

Vấn đề đối với các hệ mã khóa công khai nói chung và El Gamal nói riêng là tốc độ (do phải làm việc với các số nguyên lớn), bên cạnh đó dung lượng bộ nhớ dành cho việc lưu trữ các khóa cũng lớn. Với hệ mã El Gamal chúng ta cần gấp đôi bộ nhớ để chứa bản mã so với các hệ mã khác. Ngoài ra do việc sử dụng các số nguyên tố nên việc sinh khóa và quản lý khóa cũng khó khăn hơn với các hệ mã khối. Trên thực tế các hệ mã khóa công khai thường được sử dụng kết hợp với các hệ mã khối (mã hóa khóa của hệ mã) hoặc để mã hóa các thông tin có dung lượng nhỏ và là một phần quan trọng của một phiên truyền tin nào đó.

Thám mã đối với hệ mã El Gamal

Để thực hiện thám mã hệ mã El Gamal chúng ta cần giải bài toán Logaritm rời rạc . Ở đây chúng ta sẽ xem xét hai thuật toán có thể áp dụng để giải bài toá n này, với độ phức tạp và khả năng áp dụng khác nhau.

Thuật toán Shank

Thuật toán này còn có tên khác là thuật toán cân bằng thời gian – bộ nhớ (Time-Memory Trade Off), có nghĩa là nếu chúng ta có đủ bộ nhớ thì có thể s ử dụng bộ nhớ đó để làm giảm thời gian thực hiện của thuật toán xuống.

Input: số nguyên tố p, phần tử nguyên thủy a của Z_p^*, số nguyên y.

Output: cần tìm x sao cho $a^x \bmod p = y$.

Thuật toán:

Gọi $m = [(p-1)^{1/2}]$ (lấy phần nguyên).

Bước 1: Tính $a^{mj} \bmod p$ với $0 \le j \le m-1$.

Bước 2: Sắp xếp các cặp $(j, a^{mj} \bmod p)$ theo $a^{mj} \bmod p$ và lưu vào danh sách L_1.

Bước 3: Tính $ya^{-i} \bmod p$ với $0 \le i \le m-1$.

Bước 4: Sắp xếp các cặp $(i, ya^{-i} \bmod p)$ theo $a^{mj} \bmod p$ và lưu vào danh sách L_2.

Bước 5: Tìm trong hai danh sách L_1 và L_2 xem có tồn tại cặp $(j, a^{mj} \bmod p)$ và $(i, ya^{-i} \bmod p)$ nào mà $a^{mj} \bmod p = ya^{-i} \bmod p$ (tọa độ thứ hai của hai cặp bằng nhau).

Bước 6: $x = (mj + i) \bmod (p-1)$. Kết quả này có thể kiểm chứng từ công thức $a^{mj} \bmod p = ya^{-i} \bmod p \Rightarrow a^{mj+i} \bmod p = y \bmod p \Rightarrow x = (mj + i) \bmod (p-1)$.

Độ phức tạp của thuật toán phụ thuộc vào m = $[(p-1)^{1/2}]$, với giá trị của m, chúng ta cần tính các phần tử thuộc hai danh sách L_1 và L_2, đều là các phép toán lũy thừa phụ thuộc vào j và i, i và j lại phụ thuộc vào m nên có thể nhận thấy là thuật toán này chỉ có thể áp dụng trong những trường hợp mà p nhỏ.

Thuật toán Pohlig-Hellman

Có những trường hợp đặc biệt mà bài toán Logarithm rời rạc có thể giải quyết với độ phức tạp nhỏ hơn $\boldsymbol{O}(p^{1/2})$, chẳng hạn như khi p – 1 chỉ có các ước nguyên tố nhỏ. Một thuật toán làm việc với các trường hợp như vậy đã được Pohlig và Hellman đưa ra vào năm 1978.

Giả sử $p - 1 = 2^n$.

Gọi a là phần tử nguyên thủy của Z_p^*, p là một số lẻ và $a^{(p-1)/2} \bmod p = -1$. Gọi m là số nguyên thuộc khoảng [0, p-2] mà chúng ta cần tìm để $y = a^m \bmod p$. Giả sử m được biểu diễn thành dạng nhị phân $m = m_0 + 2m_1 + 4m_2 + \ldots + 2^{n-1}m_{n-1}$. Khi đó:

$$y^{\frac{p-1}{2}} = (a^m)^{\frac{p-1}{2}} = (a^{m_0+2m_1+2^2m_2+\ldots+2^{n-1}m_{n-1}})^{\frac{p-1}{2}} = a^{m_0\frac{p-1}{2}} = \begin{cases} 1 \text{ nếu } m_0 = 0 \\ -1 \text{ nếu } m_0 = 1 \end{cases}$$

Việc tính $y^{(p-1)/2}$ mất nhiều nhất $2[\log_2 p]$ bước và sẽ cho ta m_0. Khi xác định được $y_1 = ya^{-m_0}$, ta lặp lại thao tác tương tự để tính m_1:

$$c_1^{\frac{p-1}{4}} = (a^{m_1+2m_2+\ldots+2^{n-2}m_{n-1}})^{\frac{p-1}{2}} = a^{m_1\frac{p-1}{2}} = \begin{cases} 1 \text{ nếu } m_1 = 0 \\ -1 \text{ nếu } m_1 = 1 \end{cases}$$

Quá trình tính toán cứ thế tiếp diễn cho tới khi chúng ta tìm được m_i. Độ phức tạp của thuật toán là: $n(2[\log_2 p] + 2) \sim \boldsymbol{O}((\log_2 p)^2)$.

3.4. Các hệ mã mật dựa trên các đường cong Elliptic

Hầu hết các sản phẩm và các chuẩn sử dụng các hệ mã khóa công khai để mã hóa và chữ ký điện tử hiện nay đều sử dụng hệ mã RSA. Tuy nhiên với sự phát triển của ngành thám mã và năng lực ngày càng tăng nhanh chóng của các hệ thống máy tính, độ dài khóa để đảm bảo an toàn cho hệ mã RSA cũng ngày càng tăng nhanh chóng, điều này làm giảm đáng kể hiệu năng của các hệ thống sử dụng hệ mã RSA, đặc biệt là với các ứng dụng thương mại điện tử trực tuyến hay các hệ thống realtime đòi hỏi thời gian xử lý nhanh chóng. Gần đây một hệ mã mới đã xuất hiện và có khả năng thay thế cho RSA, đó là các hệ mã khóa công khai dựa trên các đường cong Elliptic – ECC (Elliptic Curve Cryptography).

Điểm hấp dẫn nhất của các hệ mã dựa trên các đường cong Elliptic là nó cho phép đạt được tính an toàn tương đương với RSA trong khi kích thước khóa sử dụng lại nhỏ hơn rất nhiều, làm giảm số phép tính sử dụng khi mã hóa, giải mã và do đó đạt được hiệu năng và tốc độ cần thiết. Trên lý thuyết tính an toàn của ECC không cao bằng so với RSA và cũng khó giải thích một cách dễ hiểu hơn so với RSA hay Diffie-Hellman. Cơ sở toán học đầy đủ của các hệ mã dựa trên đường cong Elliptic vượt ra ngoài phạm vi của tài liệu này, trong phần này chúng ta sẽ chỉ xem xét các vấn đề cơ bản của các đường cong Elliptic và các hệ mã ECC.

3.4.1. Nhóm Abel

Nhóm Abel G, thường được ký hiệu là {G, •} là một tập hợp với một phép toán hai ngôi ký hiệu là •, kết qủa thực hiện của phép toán với hai phần tử a, b ∈ G, ký hiệu là (a • b) cũng là một phần tử thuộc G, tính chất này gọi là đóng đối với tập G. Đối với phép toán • các mệnh đề sau đều thỏa mãn:

(A1): ∀ a, b ∈ G thì (a • b) ∈G, tính đóng (Closure)

(A2): ∀ a, b, c ∈ G thì a • (b • c) = (a • b) • c, tính kết hợp (Associate)

(A3): Tồn tại e ∈ G: e • a = a • e = a ∀ a ∈ G, e được gọi là phần tử đơn vị của tập G.

(A4): ∀ a ∈ G, luôn ∃ a' ∈ G: a • a' = a' • a = e, a' là phần tử nghịch đảo của a.

(A5): ∀ a, b ∈ G: a • b = b • a, tính giao hoán (Commutative).

Rất nhiều các hệ mã khóa công khai dựa trên các nhóm Abel. Chẳng hạn, giao thức trao đổi khóa Diffie-Hellman liên quan tới việc nhân các cặp số nguyên khác không theo modulo q (nguyên tố). Các khóa được sinh ra bởi phép tính lũy thừa trên nhóm.

Đối với các hệ mã ECC, phép toán cộng trên các đường cong Elliptic được sử dụng là phép toán cơ bản. Phép nhân được định nghĩa là sự lặp lại của nhiều phép cộng : a x k = (a + a + … + a). Việc thám mã liên quan tới việc xác định giá trị của k với các thông tin công khai là a và (a x k).

Một đường cong Elliptic là một phương trình với hai biến và các hệ số. Các đường cong sử dụng cho các hệ mã mật có các biến và các hệ thống là các phần tử thuộc về một trường hữu hạn, điều này tạo thành một nhóm Abel. Trước hết chúng ta sẽ xem xét các đường cong Elliptic trên trường số thực.

3.4.2. Các đường cong Elliptic trên trường số thực

Các đường cong Elliptic không phải là các đường Ellipse. Tên gọi đường cong Elliptic được đặt vì loại đường cong này được mô tả bởi các phương trình bậc ba, tương tự như các phương trình được dùng để tính chu vi của một Ellipse. Ở dạng chung nhất phương trình bậc 3 biểu diễn một đường cong Elliptic có dạng:

$y^2 + axy + by = x^3 + cx^2 + dx + e.$

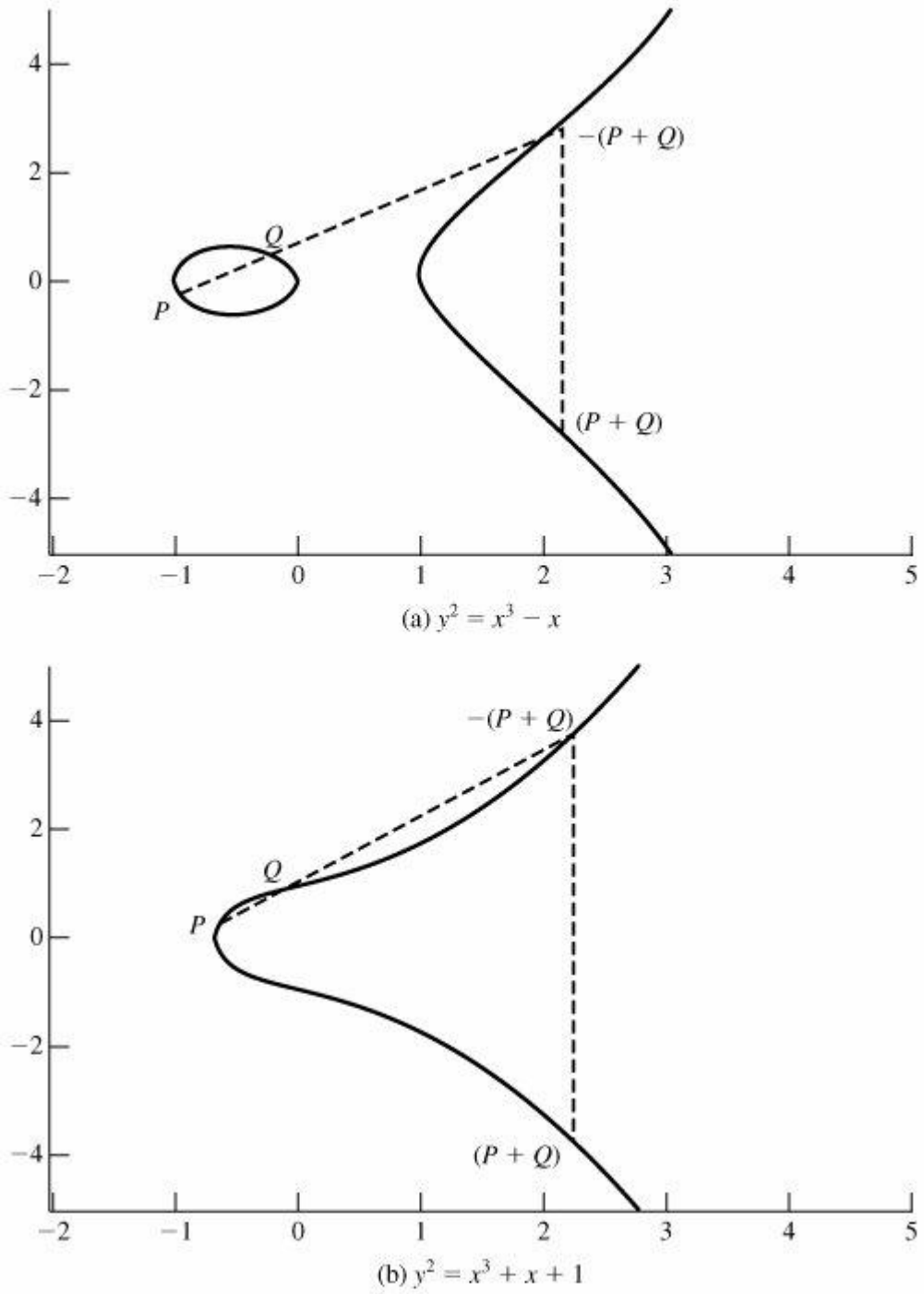
Trong đó a, b, c, d, e là các số thực, x và y là các biến thuộc trường số thực. Với mục đích để hiểu về các hệ mã ECC chúng ta chỉ xét các dạng đường cong Elliptic có dạng:

$y^2 = x^3 + ax + y$ (phương trình 1)

Các phương trình này được gọi là các phương trình bậc ba, trên các đường cong Elliptic chúng ta định nghĩa một điểm đặc biệt gọi là điểm ***O*** hay điểm tại vô cùng (point at infinity). Để vẽ đường cong Elliptic chúng ta cần tính các giá trị theo phương trình:

$$y = \sqrt{x^3 + ax + b}$$

Với mỗi giá trị cụ thể của a và b, sẽ cho chúng ta hai giá trị của y (một âm và một dương) tương ứng với một giá trị của x, các đường cong dạng này luôn đối xứng qua đường thẳng y = 0. Ví dụ về hình ảnh của một đường cong Elliptic:



Hình 4.4: Các đường cong Elliptic trên trường số thực

Chúng ta xem xét tập điểm E (a, b) chứa tất cả các điểm (x, y) thỏa mãn phương trình 1, cùng với điểm ***O***. Sử dụng các cặp (a, b) khác nhau chúng ta có các tập E (a, b) khác nhau. Sử dụng ký hiệu này ta có hình vẽ minh họa trên là biểu diễn của hai tập hợp E(1, 0) và E(1, 1) tương ứng.

3.4.3. Mô tả hình học của phép cộng trên các đường cong Elliptic

Với mỗi cặp (a, b) cụ thể chúng ta có thể thành lập một nhóm trên tập E (a, b) với các điều kiện sau:

$4a^3 + 27b^2 \neq 0$ (điều kiện 1).

Với điều kiện bổ sung này ta định nghĩa phép cộng trên đường cong Elliptic , mô tả về mặt hình học như sau: nếu ba điểm trên một đường cong Elliptic tạo thành một đường thẳng thì tổng của chúng bằng ***O***. Với định nghĩa này các luật của phép cộng trên đường cong Elliptic như sau:

1. ***O*** là phần tử trung hòa của phép cộng. $\forall$ P $\in$ E(a, b): P + ***O***= P. Trong các mệnh đề sau chúng ta giả sử P, Q ≠ ***O***.
2. P = (x, y) thì phần tử đối của P, ký hiệu là *P*, sẽ là (x, -y) và P + (*P*) = P P = *O*. P và P nằm trên một đường thẳng đứng
3. Để cộng hai điểm P và Q không có cùng hoàng độ x, vẽ một đường thẳng nối chúng và tìm giao điểm R. Dễ dàng nhận thấy chỉ có một điểm R như vậy, tổng của P và Q là điểm đối xứng với R qua đường thẳng y = 0.
4. Giao điểm của đường thẳng nối P với đối của P, tức *P*, được xem như cắt đường cong tại điểm vô cực và đó chính là ***O***.
5. Để nhân đôi một điểm Q, ta vẽ một tiếp tuyến tại Q với đường cong và tìm giao điểm S: Q + Q = 2Q = S.

Với 5 điều kiện này E(a, b) là một nhóm Abel.

3.4.4. Mô tả đại số về phép cộng

Trong phần này chúng ta sẽ trình bày một số kết quả cho phép tính toán trên các đường cong Elliptic. Với hai điểm phân biệt P = (x_P, y_P) và Q = (x_Q, y_Q) không phải là đối của nhau, độ dốc của đường nối l giữa chúng là Ä = (y_Q, y_P). Có chính xác một điểm khác mà l giao với đường cong , và đó chính là đối của tổng giữa P và Q . Sau một số phép toán đại số chúng ta có thể tính ra R = P + Q như sau:

$$x_R = \Delta^2 - y_P - x_Q$$

$$y_R = -y_P + \Delta(x_P - y_R)$$

Phép toán nhân đôi đối với P được tính như sau:

$$x_R = (\frac{3x_P^2 + a}{2y_P})^2 - 2x_P$$

$$y_R = (\frac{3x_P^2 + a}{2y_P})(x_P - x_R) - y_P$$

3.4.5. Các đường cong Elliptic trên Z_P

Các hệ mã ECC sử dụng các đường cong Elliptic với các biến và các hệ số giới hạn thuộc về một trường hữu hạn. Có hai họ các đường cong Elliptic có thể sử dụng với các hệ mã ECC: các đường cong nguyên tố trên Z_P và các đường cong nhị phân trên GF(2^m). Một đường cong nguyên tố trên Z_P, chúng ta sử dụng phương trình bậc ba mà các biến và các hệ số của nó đều là các giá trị nguyên nằm từ 0 tới p-1 và các phép tính được thực hiện theo modulo P. Trên đường cong nhị phân, các biến và các hệ số là các giá trị trên GF(2^n). và các tính toán được thực hiện trên GF (2^n). Các nghiên cứu về lý thuyết đã cho thấy các đường cong nguyên tố là phù hợp nhất cho các ứng dụng phần mềm vì những phức tạp trong tính toán đối với các đường cong nhị phân, nhưng đối với các ứng dụng phần cứng thì việc sử dụng các đường cong nhị phân lại tốt hơn vì cơ chế làm việc của các mạch, các con chíp rất phù hợp với các tính toán trên trường nhị phân.

Với các đường cong Elliptic trên Z_P chúng ta định nghĩa lại phương trình biểu diễn như sau:

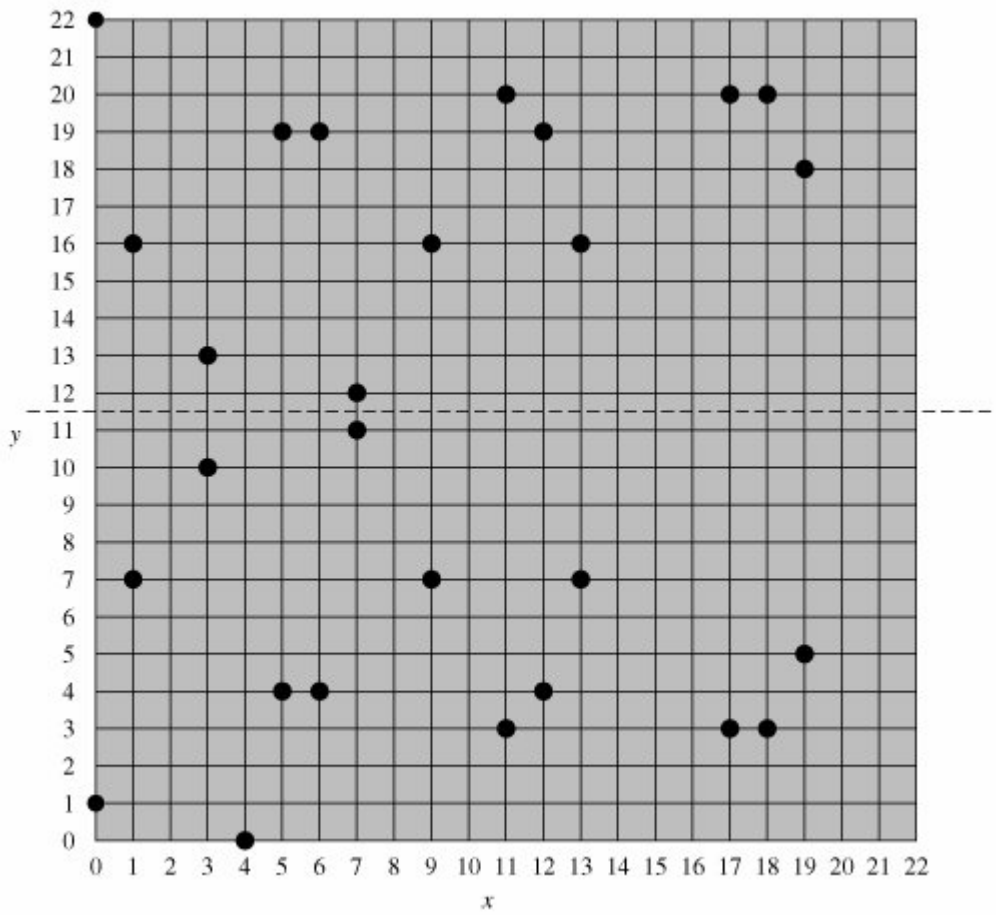
$y^2 \bmod p = (x^3 + ax + y) \bmod p$. (phương trình 2)

Chẳng hạn các giá trị a = 1, b = 1, x = 9, y = 9, y = 7, p = 23 thỏa mãn phương trình trên.

Các giá trị hệ số a, b và các biến số x, y đều thuộc Z_P. Tập $E_P(a, b)$ gồm tất cả các cặp (x, y) thỏa mãn phương trình phương trình 2.

Ví dụ với p = 23, a = b = 1, ta có tập $E_{23}(1, 1)$:

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)



Bảng 4.2: Biểu diễn của tập E23(1, 1)

Các qui tắc về phép cộng cũng được định nghĩa tương tự đối với các đường cong Elliptic nguyên tố:

Điều kiện: $(4a^3 + 27b^2) \bmod p \neq 0$.

1. $P + \boldsymbol{O} = P$

2. Nếu $P = (x_P, y_P)$ thì $P + (x_P, y_P) = O$, điểm (x_P, y_P) được gọi là đối của P , ký hiệu là *P*. Chẳng hạn trên $E_{23}(1, 1)$, P = (13, 7) ta có *P* = (13, 7) nhưng 7 mod 23 = 16 nên *P* = (13, 16), cũng thuộc $E_{23}(1, 1)$.

3. Với hai điểm phân biệt $P = (x_P, y_P)$ và $Q = (x_Q, y_Q)$, $R = P + Q = (x_R, y_R)$ được định nghĩa như sau:

$$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$

$$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$$

Trong đó:

$$\lambda = \begin{cases} (\frac{y_Q - y_P}{x_Q - x_P}) \bmod p, (P \neq Q) \\ (\frac{3x_P^2 + a}{2y_P}) \bmod p, () p = Q) \end{cases}$$

4. Phép nhân được định nghĩa là tổng của các phép cộng , chẳng hạn 4P = P + P + P + P. Ví dụ với P = (3, 10) và Q = (9, 7) trên $E_{23}(1, 1)$ ta có:

$$\lambda = (\frac{7-10}{9-3}) \bmod 23 = (\frac{-3}{6}) \bmod 23 = (\frac{-1}{2}) \bmod 23 = 11 \text{ nên}$$

$x_R = (11^2 - 3 - 9) \bmod 23 = 17$

$y_R = (11(3 - 17) - 10) \bmod 23 = 20$. Nên P + Q = (17, 20).

Để tìm 2P ta tính:

$$\lambda = (\frac{3(3^2)+1}{2 \times 10}) \bmod 23 = (\frac{5}{20}) \bmod 23 = (\frac{1}{4}) \bmod 23 = 6$$

Chú ý là để thực hiện phép tính cuối cùng ta lấy phần tử nghịch đảo của 4 trên Z_{23} sau đó nhân với tử số là 1.

$x_R = (6^2(3 - 7) - 10) \bmod 23 = 30 \bmod 23 = 7$

$y_R = (6(3 - 7) - 10) \bmod 23 = 34 \bmod 23 = 12$

Kết luận: 2P = (7, 12).

Để xác định độ an toàn của các hệ mã mật dựa trên các đường cong Elliptic , người ta thường dựa trên một con số là số phần điểm trên mộ t nhóm Abel hữu hạn , gọi là N , được định nghĩa trên một đường cong Elliptic . Trong trường hợp nhóm hữu hạn $E_P(a, b)$, ta có các cận của N là:

$p+1-2\sqrt{p} \leq N \leq p+1+2\sqrt{p}$, con số này xấp xỉ bằng số phần tử của Z_P (bằng p).

3.4.6. Các đường cong Elliptic dựa trên các trường hữu hạn GF(2^m)

Số phần tử của trường hữu hạn GF (2^m) là 2^m, các phép toán được trang bị trên GF(2^m) là phép toán cộng và phép toán nhân được thực hiện với các đa thức . Đối với các đường cong Elliptic dựa trên GF (2^m), chúng ta sử dụng một phương trình bậc ba với các biến và các tham số có giá trị thuộc GF (2^m), các phép tính được thực hiện tuân theo các phép toán trên GF(2^m).

1. Phương trình biểu diễn

So với các hệ mã mật dựa trên các đường cong trên Z_p, dạng biểu diễn của các hệ mã dựa trên GF(2^m) tương đối khác:

$y^2 + xy = x^3 + ax^2 + b$ (phương trình 3)

Trong đó các biến x, y và các hệ số a, b là các phần tử của GF(2^m) và các phép tính toán được thực hiện tuân theo các qui tắc trên GF(2^m).

Chúng ta ký hiệu $E_2^m(a, b)$ là tất cả các cặp số nguyên (x, y) thỏa mãn phương trình phương trình 3 và điểm vô cùng ***O***.

Ví dụ: chúng ta có thể sử dụng GF(2^4) với đa thức bất khả qui $f(x) = x^4 + x + 1$. Phần tử sinh của GF(2^4) là g thỏa mãn f(g) = 0, $g^4 = g + 1$, hay ở dạng nhị phân là 0010. Chúng ta có bảng lũy thừa của g như sau:

g0 = 0001	g4 = 0011	g8 = 0101	g12 = 1111
g1 = 0010	g5 = 0110	g9 = 1010	g13 = 1101
g2 = 0100	g6 = 1100	g10 = 0111	g14 = 1001
g3 = 1000	g7 = 1011	g11 = 1110	g15 = 0001

Chẳng hạn $g^5 = g^4 g = (g+1)g = g^2 + g = 0110$.

Xét đường cong Elliptic $y^2 + xy = x^3 + g^4x^2 + 1$, trong trường hợp này $a = g^4$ và $b = g^0 = 1$. Một điểm nằm trên đường cong là (g^5, g^3):

$(g^3)^2 + (g^5)(g^3) = (g^5)^3 + (g^4)(g^5)^2 + 1$

$\Leftrightarrow g^6 + g^8 = g^{15} + g^{14} + 1$

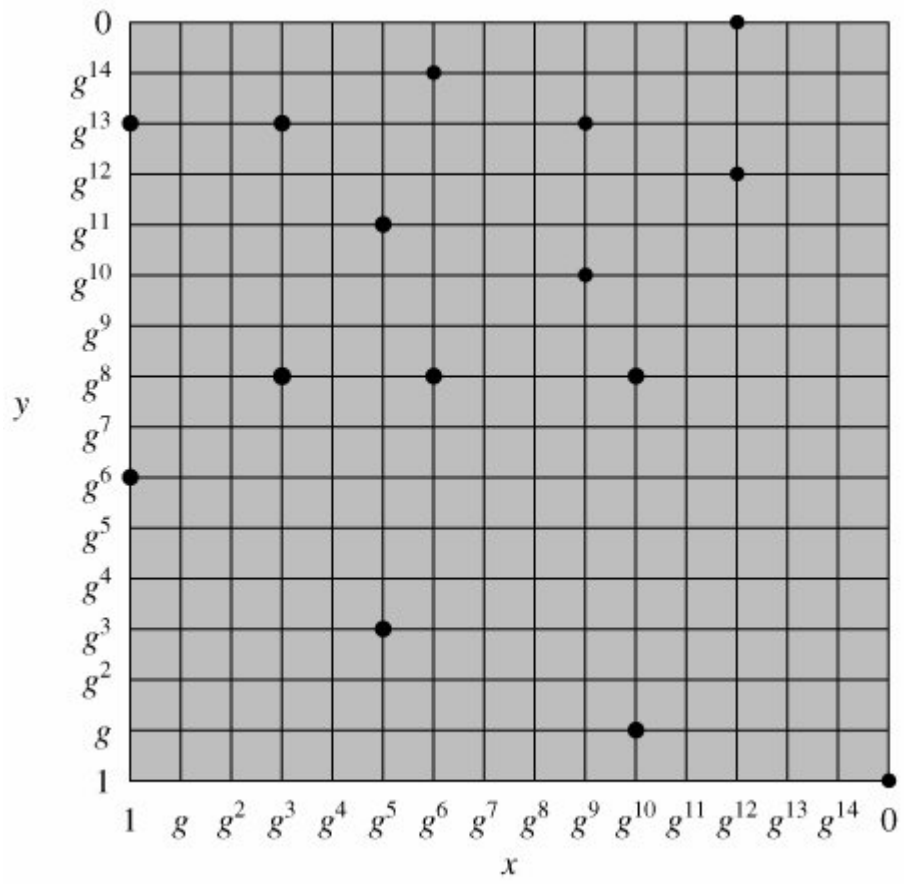
$\Leftrightarrow 1100 + 0101 = 0001 + 1001 + 0001$

$\Leftrightarrow 1001 = 1001$

Bảng sau là các điểm trên $E_2^4(g^4, 1)$:

(0, 1)	(g^5, g^3)	(g^9, g^{13})
$(1, g^6)$	(g^5, g^{11})	(g^{10}, g)
$(1, g^{13})$	$g^6, g^8)$	(g^{10}, g^8)
(g^3, g^8)	(g^6, g^{14})	$(g^{12},0)$
(g^3, g^{13})	(g^9, g^{10})	(g^{12}, g^{12})

Hình biểu diễn tương đương:



Hình 4.5: Hình biểu diễn $E_2^4(g^4, 1)$

Một nhóm Abel có thể định nghĩa dựa trên $E_2^m(a, b)$ với điều kiện b≠0. Các luật thực hiện với phép cộng, ∀ a, b ∈$E_2^m(a, b)$:

1. P + ***O*** = P
2. Nếu P = (x_P, y_P) thì P + (x_P, x_P + y_P) = ***O***. Điểm (x_P, x_P + y_P) là điểm đối của P, ký hiệu là *P*.
3. Nếu P = (x_P, y_P) và Q = (x_Q, y_Q) và P≠Q, *P*≠Q thì R = P + Q = (x_R, y_R) được xác định bằng các công thức sau:

$$x_R = \lambda^2 + \lambda + x_P + x_Q + a$$

$$y_R = \lambda(x_P + x_R) + x_R + y_P + a$$

Trong đó:

$$\lambda = \frac{y_Q + y_P}{x_Q + x_P}$$

4. Nếu P = (x_P, y_P) thì R = 2P = (x_R, y_R) được xác định bằng các công thức sau:

$$x_R = \lambda^2 + \lambda + a$$

$$y_R = x_P^2 + (\lambda + 1)x_R$$

Trong đó:

$$\lambda = x_P + \frac{y_P}{x_P}$$

3.4.7. Hệ mã mật dựa trên các đường cong Elliptic

Phép toán cộng trên đường cong Elliptic tương ứng với phép nhân theo modulo trong hệ mã RSA, còn phép toán nhân (cộng nhiều lần) trên đường cong Ellipti c tương ứng với phép lũy thừa theo modulo trong hệ mã RSA. Tương tự như bài toán cơ sở của hệ mã RSA là bài toán phân tích ra dạng thừa số nguyên tố của một số nguyên lớn, các hệ mã dựa trên các đường cong Elliptic cũng có các bài toán cơ sở là một bài toán khó giải, gọi là bài toán Logarithm trên đường cong Elliptic:

Xét phương trình Q = kP trong đó P, Q $\in E_P(a, b)$ và k < p. Việc tính Q nếu biết P và k là một bài toán dễ (thực hiện theo các công thức). Nhưng việc xác định k với giá trị P, Q cho trước lại là bài toán khó.

Chúng ta xem xét ví dụ (Certicom Website www.certicom.com): $E_{23}(9, 17)$ được xác định bởi phương trình $y^2 \bmod 23 = (x^3 + 9x + 17) \bmod 23$.

Với Q = (4, 5) và P = (16, 5) thì k thỏa mãn Q = kP sẽ bằng bao nhiêu ? Phương pháp đơn giản nhất là nhân P lên nhiều lần cho tới khi bằng Q:

P = (16, 5), 2P = (20, 20), 3P = P = (16, 5); 2P = (20, 20); 3P = (14, 14); 4P = (19, 20); 5P = (13, 10); 6P = (7, 3); 7P = (8, 7); 8P (12, 17); 9P = (4, 5).

Như vậy k = 9. Trên thực tế các hệ mã sẽ đảm bảo giá trị k là đủ lớn để phương pháp vét cạn như trên là không thể thực hiện được.

3.4.8. Phương pháp trao đổi khóa Diffie-Hellman dựa trên các đường cong Elliptic

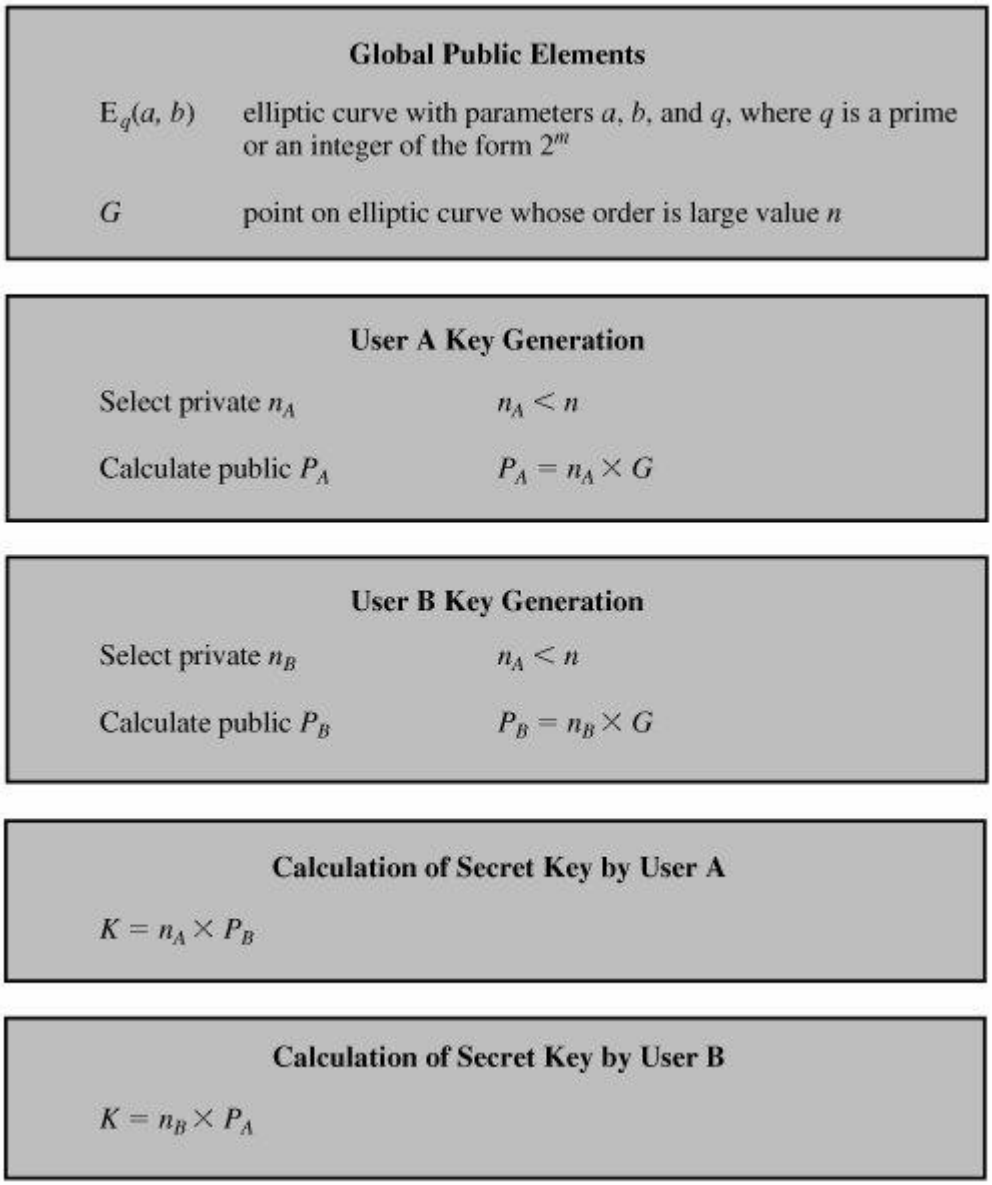
Ban đầu người ta chọn một số nguyên lớn q, có thể là một số nguyên tố p hay có dạng 2^m tương ứng với các phương trình biểu diễn và các tham số a, b. Việc lựa chọn này cho chúng ta tập hợp $E_q(a, b)$. Tiếp theo chọn một điểm $G = (x_1, y_1) \in E_P(a, b)$ có bậc n rất lớn, bậc n của điểm G là số nguyên nhỏ nhất thỏa mãn nG = ***O***. $E_q(a, b)$ và G là các tham số công khai cho hệ mã mật dựa trên đường cong Elliptic tương ứng với các tham số p, a, b.

Phương pháp trao đổi khóa giữa hai người dùng A và B có thể thực hiện như sau:

1. A chọn một số nguyên n_A nhỏ hơn n. Đó chính là khóa riêng của A. Sau đó sinh khóa công khai $P_A = n_A \times G$, khóa này là một điểm trên $E_q(a, b)$.
2. Tương tự B cũng chọn một khóa riêng n_B và tính khóa công khai P_B.
3. A sinh một khóa bí mật $K = n_A \times P_B$. B sinh khóa bí mật $K = n_B \times P_A$.

Dễ dàng kiểm chứng các khóa bí mật của A và B tính được đều bằng nhau: $n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$.

Hình minh họa các bước:



Hình 4.6: Phương pháp trao đổi khóa Diffie-Hellman dựa trên ECC

Để tấn công phương pháp trao đổi khóa trên , kẻ tấn công cần phải tính được giá trị k với các giá trị công khai là G và kG, và đây chính là bài toán Logarithm trên đường cong Elliptic, một bài toán khó.

Ví dụ: p = 211, $E_{211}(0, 4)$ tương ứng với phương trình biểu diễn $y^2 = x^3 + 4$, ta chọn G = (2, 2). Do 240G = ***O*** nên n = 240. A chọn khóa riêng là $n_A = 121$, khóa công khai tương ứng của A sẽ là $P_A = 121(2, 2) = (115, 48)$. Khóa riêng của B là $n_B = 203$ nên khóa công khai cùa B là $P_B = 203(2, 2) = (130, 203)$. Khóa bí mật (chia sẻ) giữa A và B là 121(130, 203) = 203(115, 48) = (161, 69).

3.4.9. Thuật toán mã hóa và giải mã

Có nhiều cách mã hóa /giải mã đã được nghiên cứu với các hệ mã trên các đường cong Elliptic, ở đây chúng ta sẽ xem xét cách đơn giản nhất. Thuật toán mã hóa ban đầu sẽ thực hiện phép biến đổi tiền xử lý từ input là một bản rõ m thành dạng một điểm P_m. Điểm P_m sẽ được mã hóa thành bản mã và sau đó giải mã . Thực chất việc tiền xử lý này không đơn giản vì không phải tất cả các tọa độ có dạng (x, y) đều thuộc $E_P(a, b)$. Có

nhiều cách khác nhau cho việc tiền xử lý này , chúng ta không bàn kỹ tới chúng ở đây nhưng thực tế là có một vài cách dễ hiểu để thực hiện việc đó.

Giống như đối với hệ trao đổi khóa , chúng ta cần một điểm G và một nhóm Elliptic $E_q(a, b)$ làm tham số . Mỗi người dùng A lựa chọn một khóa riêng n_A và sinh một khóa công khai $P_A = n_A \times G$.

Để mã hóa một thông điệp P_m để gửi tới cho B , A sẽ chọn một số nguyên dương ngẫu nhiên k và sinh bản mã C_m gồm một cặp điểm:

$C_m = \{kG, P_m + kP_B\}$.

Chú ý là ở đây A sử dụng khóa công khai của B . Để giải mã bản mã , B sẽ nhân điểm thứ nhất với khóa bí mật của B và lấy kết quả nhận được trừ đi điểm thứ hai:

$P_m + kP_B n_B(kG) = P_m + k(n_BG) n_B(kG) = P_m$.

A đã che đi giá trị của P_m bằng cách cộng kP_B vào P_m. Chỉ có duy nhất A biết giá trị k, nên thậm chí biết khóa công khai P_B, không ai có thể loại bỏ mặt nạ kP_B để tìm ra P_m. Tuy nhiên giá trị của C_m cũng gồm một đầu mối để B (người duy nhất giữ khóa riêng n_B) có thể dựa vào đầu mối đó mà tìm ra P_m.

Ví dụ: p = 751, $E_P(1, 188)$ tương ứng với phương trình $y^2 = x^3 + x + 188$, G = (0, 376). Giả sử A muốn gửi một thông điệp tương ứng với P_m = (562, 201) và A lựa chọn k = 386, khóa công khai của B là P_B = (201, 5). Chúng ta có 386(0, 376) = (676, 558) và (562, 201) + 386(201, 5) = (385, 328). Bản mã sẽ là C_m = {(676, 558), (385, 328)}.

3.4.10. Độ an toàn của các hệ mã mật dựa trên các đường cong Elliptic

Độ an toàn của các hệ mã ECC phụ thuộc vào việc xác định được giá trị của k dựa trên các giá trị kP và P. Bài toán này được gọi là bài toán Logarithm trên các đường cong Elliptic. Thuật toán nhanh nhất để giải bài toán này là thuật toán của Pollard . Bảng sau cho chúng ta sự so sánh tương quan giữa các hệ mã:

Symmetric Scheme (key size in bits)	ECC-Based Scheme (size of n in bits)	RSA/DSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
92	384	7680
256	512	15360
Nguồn: Certicom		

Bảng 4.3: Bảng so sánh các hệ mã ECC với hệ mã RSA

Có thể thấy là so với RSA , các hệ mã ECC có ưu thế hơn về độ dài khóa sử dụng , đặc biệt là khi chúng ta sử dụng các khóa có độ dài nhỏ thì ECC còn có ưu thế về tốc độ (số phép tính) xử lý trong mã hóa và giải mã.

4. Bài tập

Bài tập 4.1: Cho N = 1517. Hãy tính 13^{1435} mod N.

Bài tập 4.2: Trong hệ mã RSA có N = p * q = 103 * (2^{19} – 1) thì có thể sử dụng tối đa là bao nhiêu gía trị của e để làm khóa mã hóa, giải thích.

Bài tập 4.3: Trong hệ mã RSA có N = p*q = 103 * 113 sẽ có bao nhiêu trường hợp lộ bản rõ.

Bài tập 4.4: Trong hệ chữ ký điện tử ElGamma có p = 2^{31} – 1 khi ký lên một văn bản có thể sử dụng tối đa bao nhiêu gía trị k, giải thích.

Bài tập 4.5: Cho hệ mã ElGamma có p = 31, a = 11 và x = 6. Để mã hóa M = 18 người ta chọn k = 7. Hãy thực hiện tính toán và đưa ra bản mã kết quả.

Bài tập 4.6: Cho hệ RSA có n = 1363, biết phi(n) = 1288 hãy mã hóa bản rõ M = 2007.

Bài tập 4.7: Tương tự Câu 1 với n = 215629 và phi(n) = 214684 hãy giải mã bản mã M = 2007.

Bài tập 4.8: Giả sử có 4 tổ chức sử dụng 4 hệ mã RSA để truyền thông với nhau. Gọi N_1, N_2, N_3, N_4 lần lượt là các tham số tương ứng mà họ sử dụng và $(N_i, N_j) = 1 \ \forall \ i \neq j$ và $i, j \in Z_5/\{0\}$. Cả bốn hệ RSA này đều có số mũ lập mã là e = 3. Một thông điệp m sau khi mã hóa bằng 4 hệ mã trên nhận được 4 bản mã tương ứng là C_1, C_2, C_3, C_4. Hãy tìm m.

Bài tập 4.9: Cho hệ mã Knapsack có A = {11, 15, 30, 60}, M = 150 và u = 77.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ mã trên.
b) Để mã hóa các thông điệp viết bằng tiếng Anh người ta dùng một hàm chuyển đổi từ các ký tự thành các xâu nhị phân như sau:

Ký tự	Xâu bít	Ký tự	Xâu bít	Ký tự	Xâu bít	Ký tự	Xâu bít
A	00000	H	00111	O	01110	V	10101
B	00001	I	01000	P	01111	W	10110
C	00010	J	01001	Q	10000	X	10111
D	00011	K	01010	R	10001	Y	11000
E	00100	L	01011	S	10010	Z	11001
F	00101	M	01100	T	10011		
G	00110	N	01101	U	10100		

Khi đó ví dụ xâu ABCD sẽ được chuyển thành 00000 00001 00010 00011 và cắt thành các xâu có độ dài 4 để thực hiện mã hóa. Kết quả thu được bản mã là một dãy các số $\in Z_M$. Hãy thực hiện mã hóa xâu P = “ANTI”.

c) Giả sử bản mã thu được là C = <120, 105, 105, 0, 60, 75, 30, 22, 22, 30>. Hãy thực hiện giải mã bản mã trên để thu được thông điệp ban đầu.

Bài tập 4.10: Cho hệ mã Knapsack có A = {7, 13, 31, 53}, M = 173 và u = 97.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ mã trên.